**Req 70- node.js/PHP Developer**

**Job Description- node.js/PHP Developer  
  
ESSENTIAL POSITION RESPONSIBILITIES**

* Design and write high quality, high performance web based applications.
* Participate in development life cycle activities like release planning, coding, testing and production release.
* Work closely with product managers for feature requirements and deliver highest quality features.
* Participate in fixing production issues.

**DESIRED SKILLS AND EXPERIENCE**

* BS/MS degree, preferably in a Computer Science or related field.
* Proven working experience in software development.
* Proficiency in leading edge front end technologies similar to ReactJS, Angular, Sass/Less, HTML5, etc.
* Working familiarity and understanding of the back end, i.e. REST APIs, SQL/NoSQL Databases, using a language such as Python/Ruby/Java/Go/Scala/NodeJS/or similar experience with remote data via RESTful APIs and JSON.
* Solid understanding of the full web development life cycle.
* Passion for designing and engineering rich user interfaces.
* Disciplined approach to testing and quality assurance.
* Hands-on knowledge of version control systems such as GIT
* Ability to think logically, diagnose/triage.
* Driven to execute and follow through.
* Experience with standard practices like TDD, Scrum/Agile is a plus.
* Experience with client side JavaScript framework such as React and Angular is a plus.
* Knowledge of HTML, CSS, JQuery is a plus

**Interview assignment:** Title: Mini Twitter Coding Challenge

Create the backend for a mini messaging service, inspired by Twitter. It should have a RESTful API where all endpoints require [HTTP Basic authentication](https://url.emailprotection.link/?a30fJDafXi4T9U2d2jYlCLZ6aJe5dEFL2F031aPSTTgeDnKKueSm0h0YIgVeQehfbtGsovZVPSJdf6S3nxvMwWg~~) and generate output in JSON format. Implement the the following basic functionality:

1. An endpoint to read the message list for the current user (as identified by their HTTP Basic authentication credentials). Include messages they have sent and messages sent by users they follow. Support a “search=” parameter that can be used to further filter messages based on keyword.
2. Endpoints to get the list of people the user is following as well as the followers of the user.
3. An endpoint to start following another user.
4. An endpoint to unfollow another user.

5. Use an in memory database like H2 / SQLite for storing data.

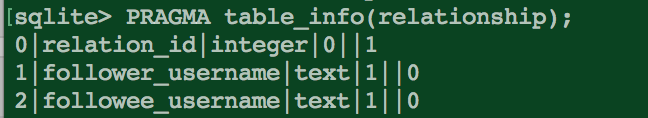
In addition to the basic functionality above, **choose ONE**of the following extra features to implement:

1. An endpoint that returns the current user's "shortest distance" to some other user. The shortest distance is defined as the number of hops needed to reach a user through the users you are following (not through your followers; direction matters). For example, if you follow user B, your shortest distance to B is 1. If you do not follow user B, but you do follow user C who follows user B, your shortest distance to B is 2.
2. An endpoint that returns a list of*all* users, paired with their most "popular" follower. The more followers someone has, the more "popular" they are. Hint: this is possible to do with a single SQL query!
3. Write a small web interface for your API in JavaScript. Don't use server-side templates (like JSPs). Instead, write a single-page application that interacts with your API via AJAX calls. Feel free to use any modern JavaScript frameworks (e.g., AngularJS, ReactJS, etc.).

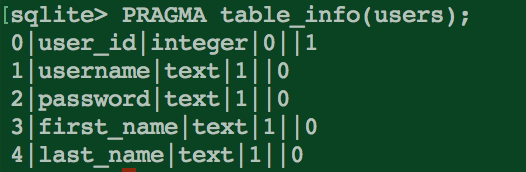
Note that the main part of your challengeis to create the *backend*for a web application. This service should work without any frontend (HTML or JavaScript). You aren't expected to write a frontend at all unless you pick option 3 above.

**Below are the assumptions and preloaded data descriptions for this RESTful app:**

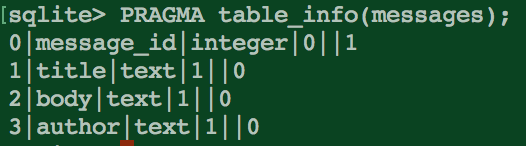
* 1. **Per the requirements, created three tables: users, relationship, messages. (file: twitter.db)**- relationship



- users



- messages

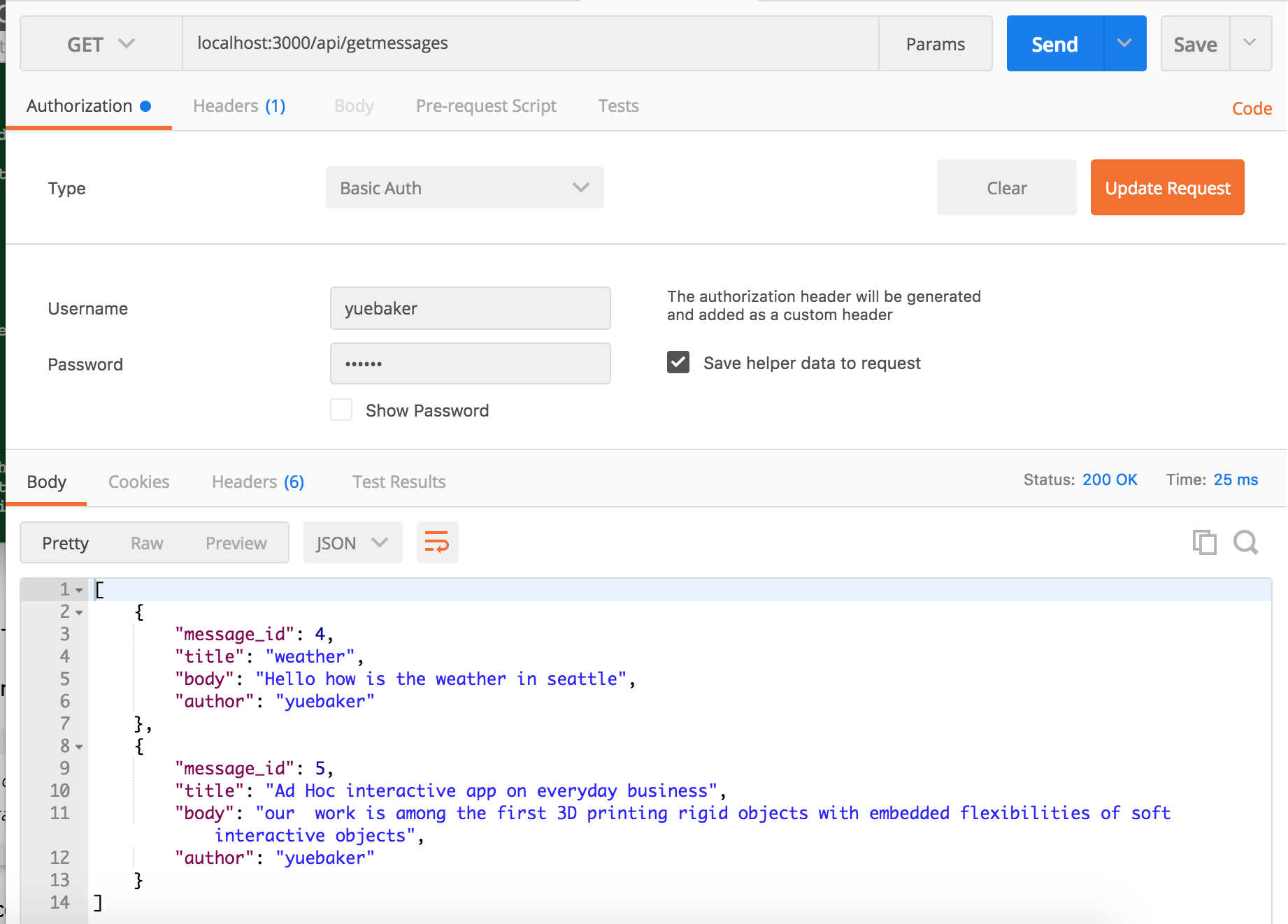


* 1. To start the app: node www

Since this is an express framework, the default server listen code is included in the www js file.

* 1. API design
* An endpoint to read the message list for the current user (as identified by their HTTP Basic authentication credentials). Include messages they have sent and messages sent by users they follow. Support a “search=” parameter that can be used to further filter messages based on keyword.

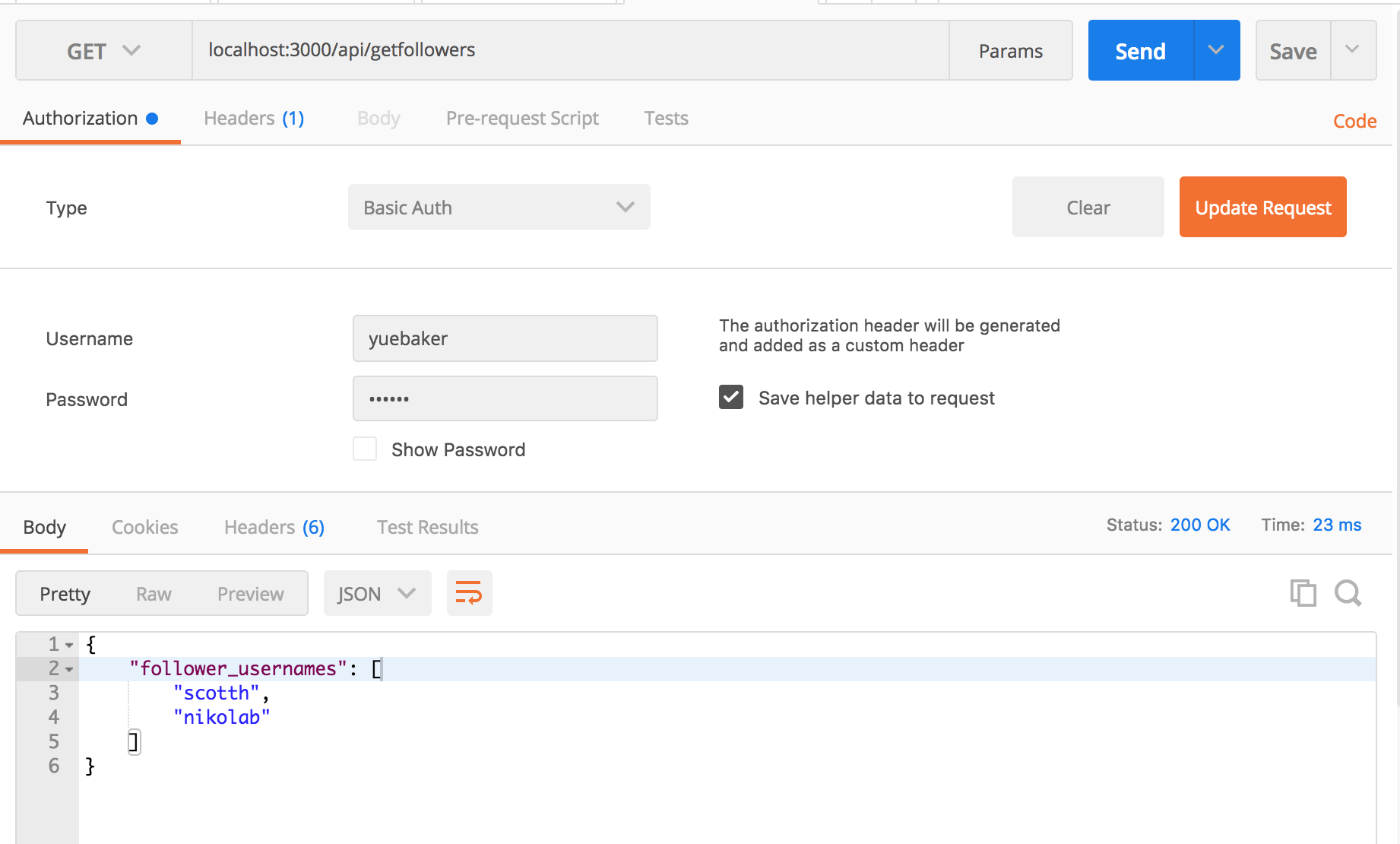
‘/getmessages’



* Endpoints to get the list of people the user is following as well as the followers of the user.

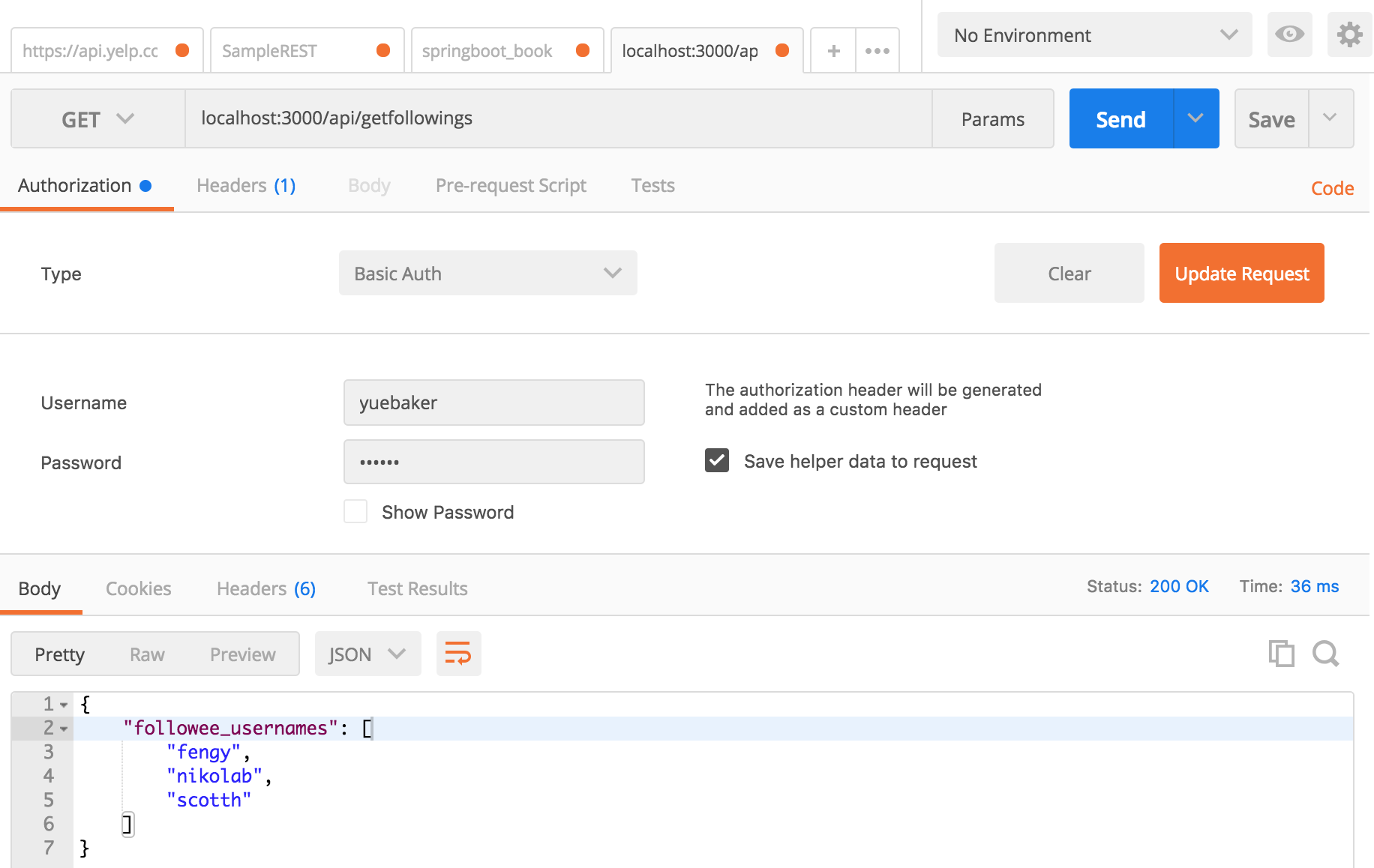
This is the api call to get a list of users which are following the current user

'/getfollowers'



This is the api call to get a list of users which the current user is following.

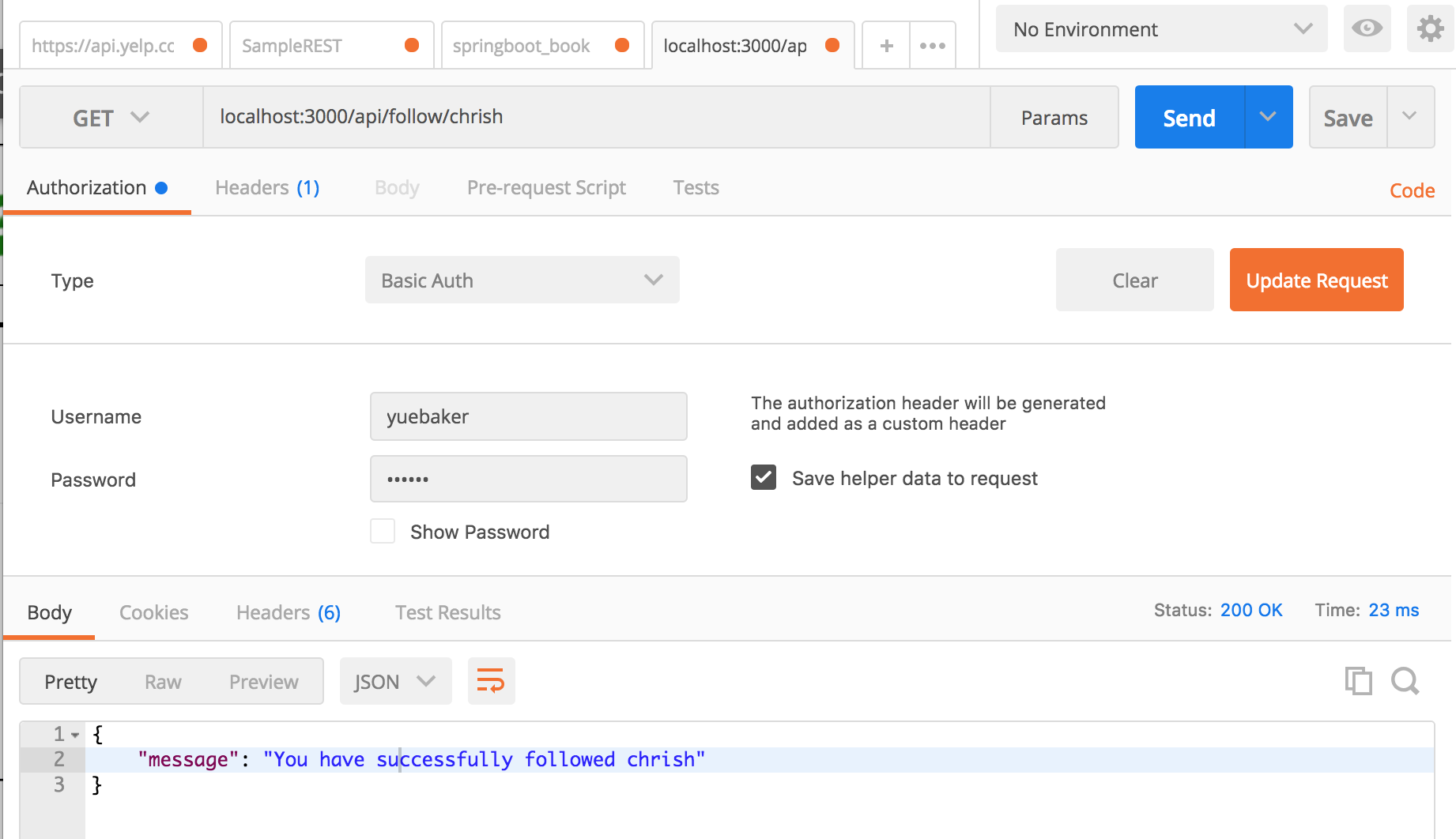
'/getfollowings’

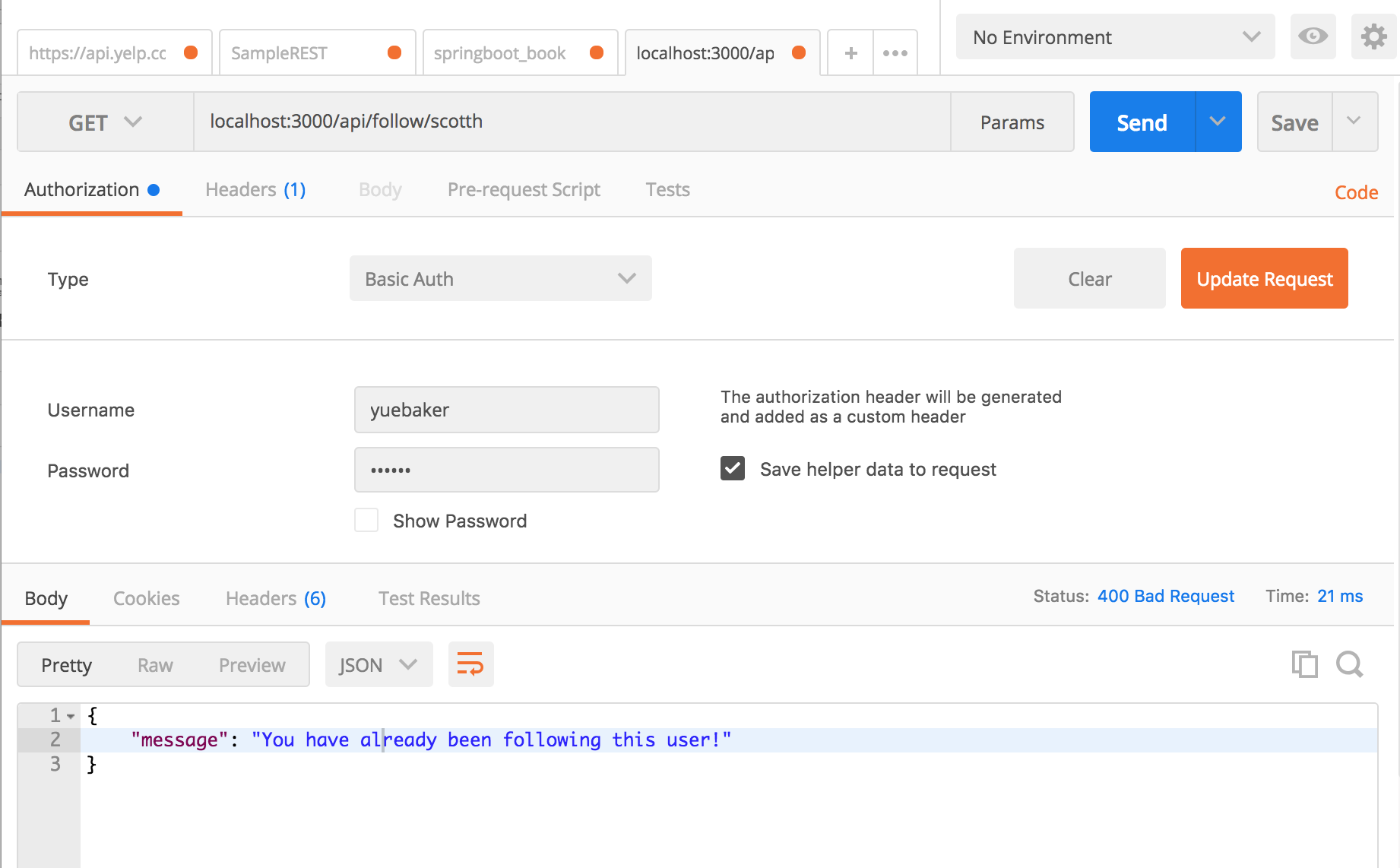


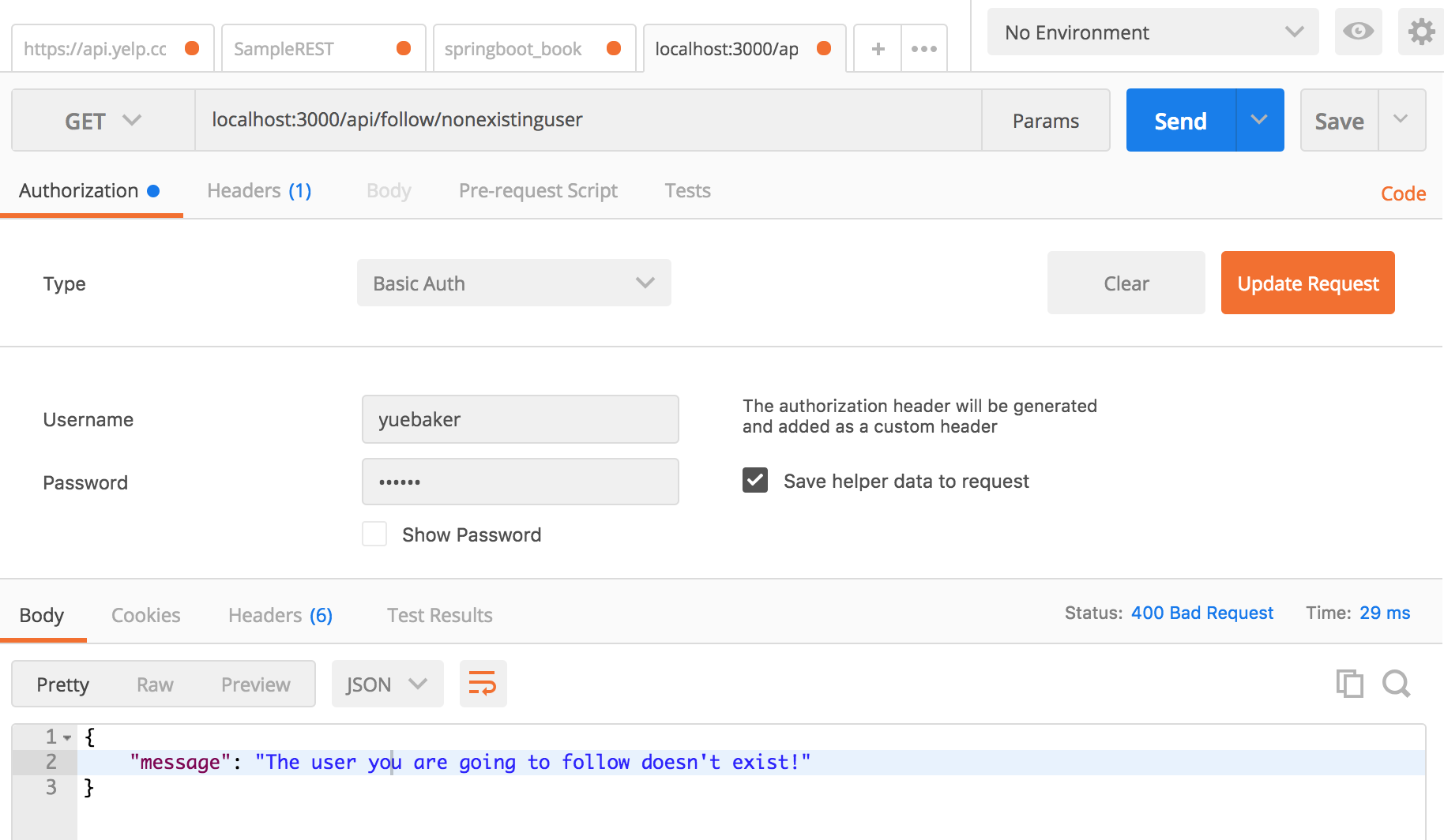
* An endpoint to start following another user.

This is the api call for start following another user.

'/follow/:username'



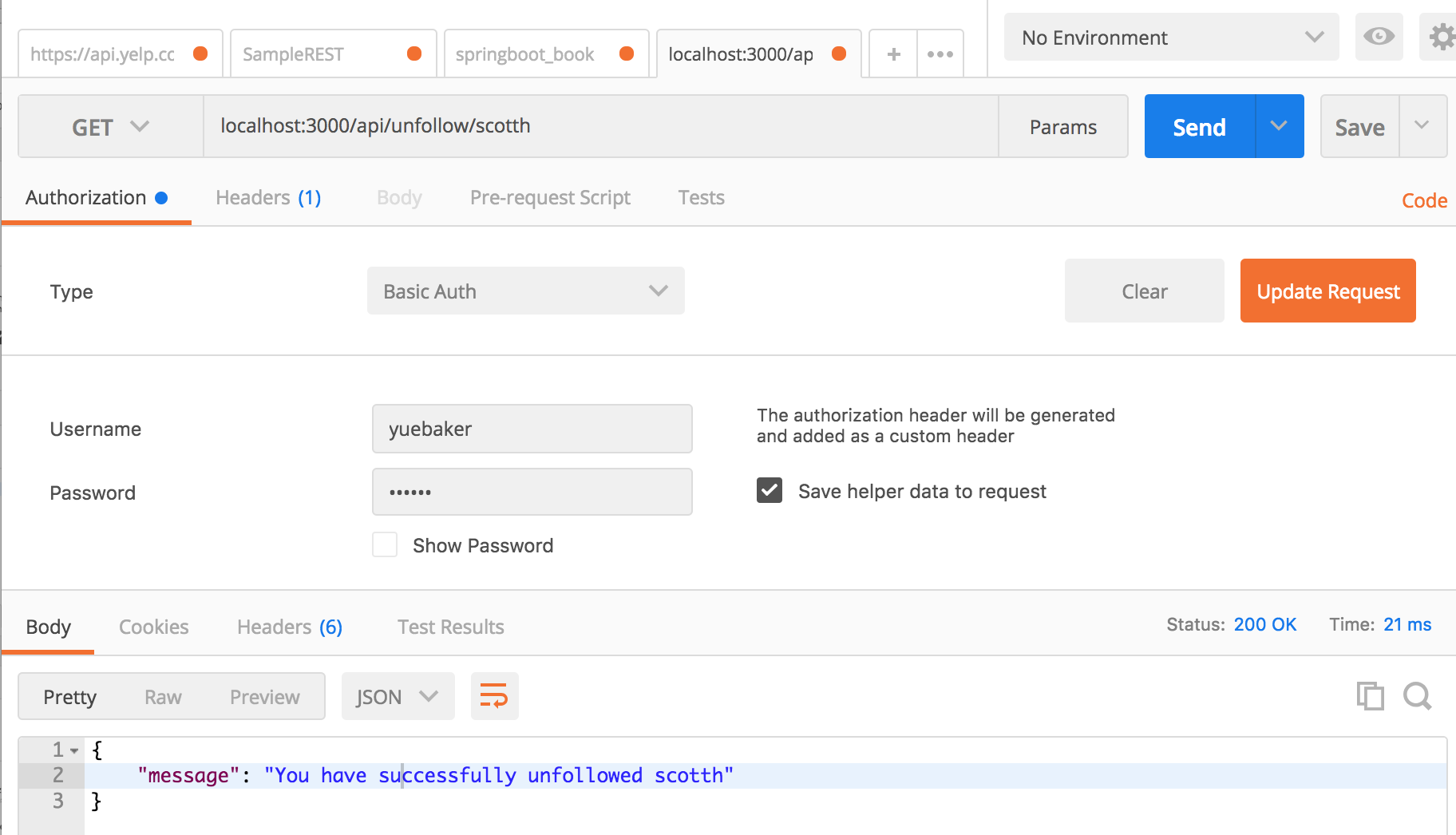


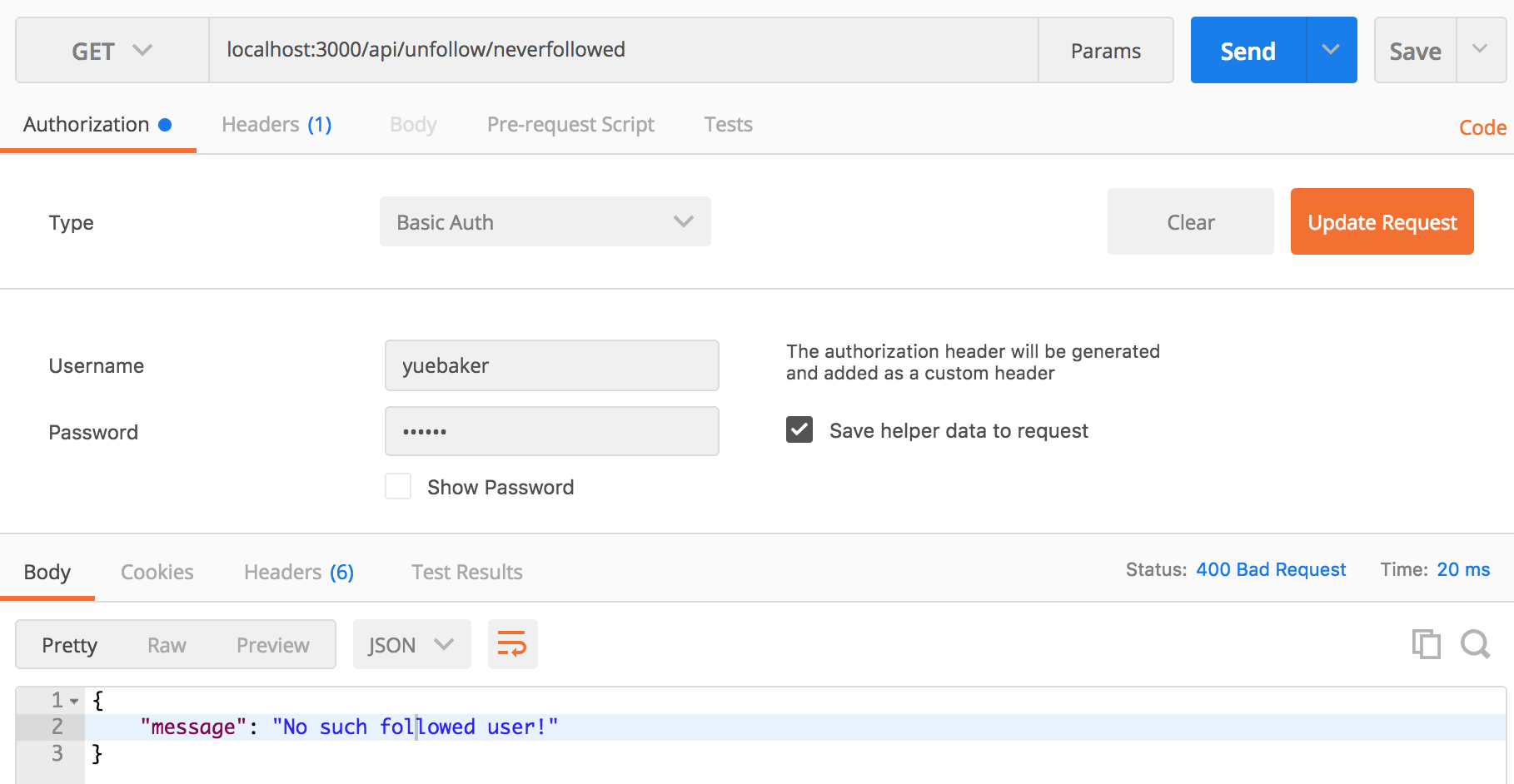


* An endpoint to unfollow another user.

This is the api call for start unfollowing another user.

'/unfollow/:username'

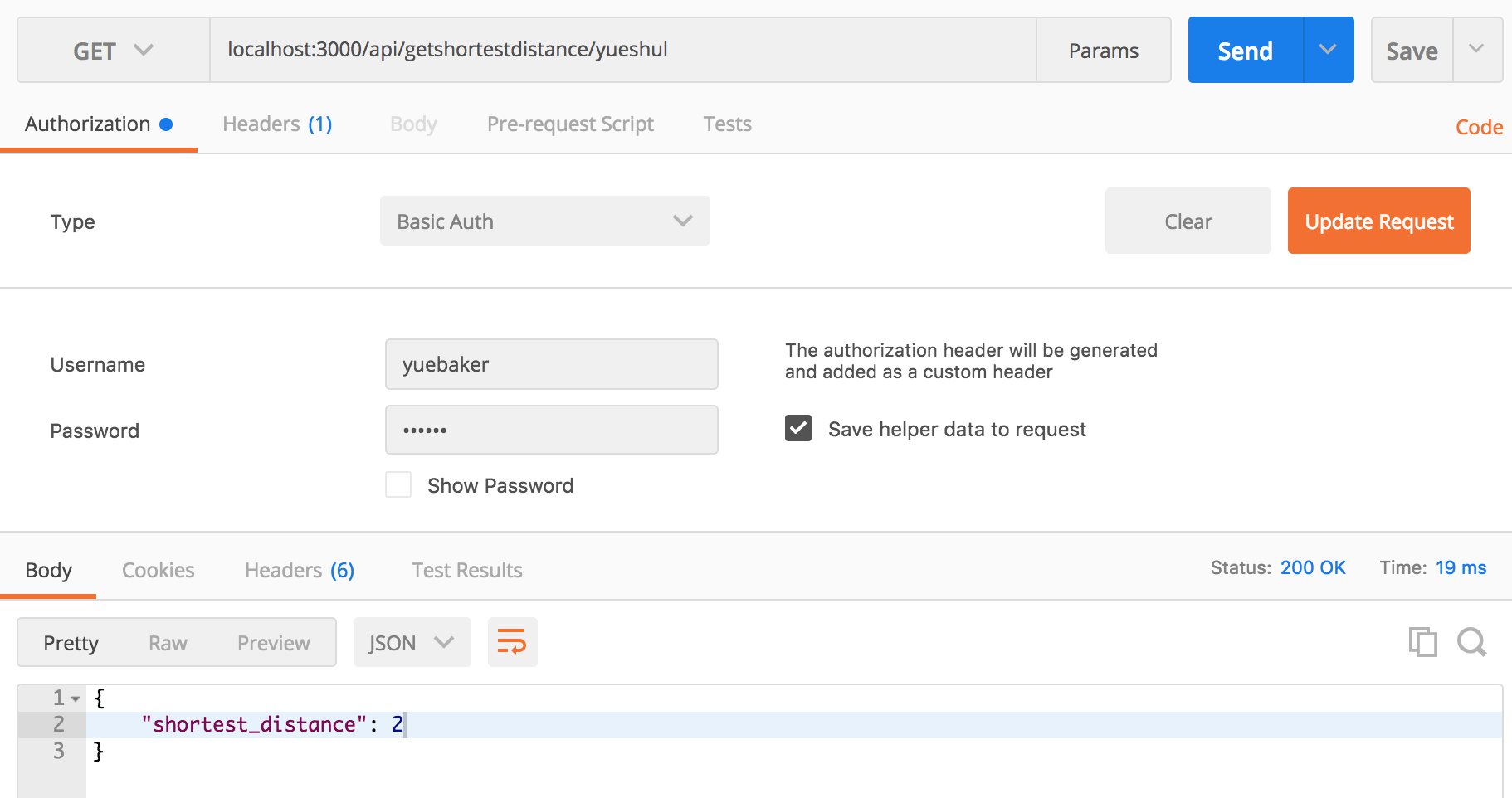


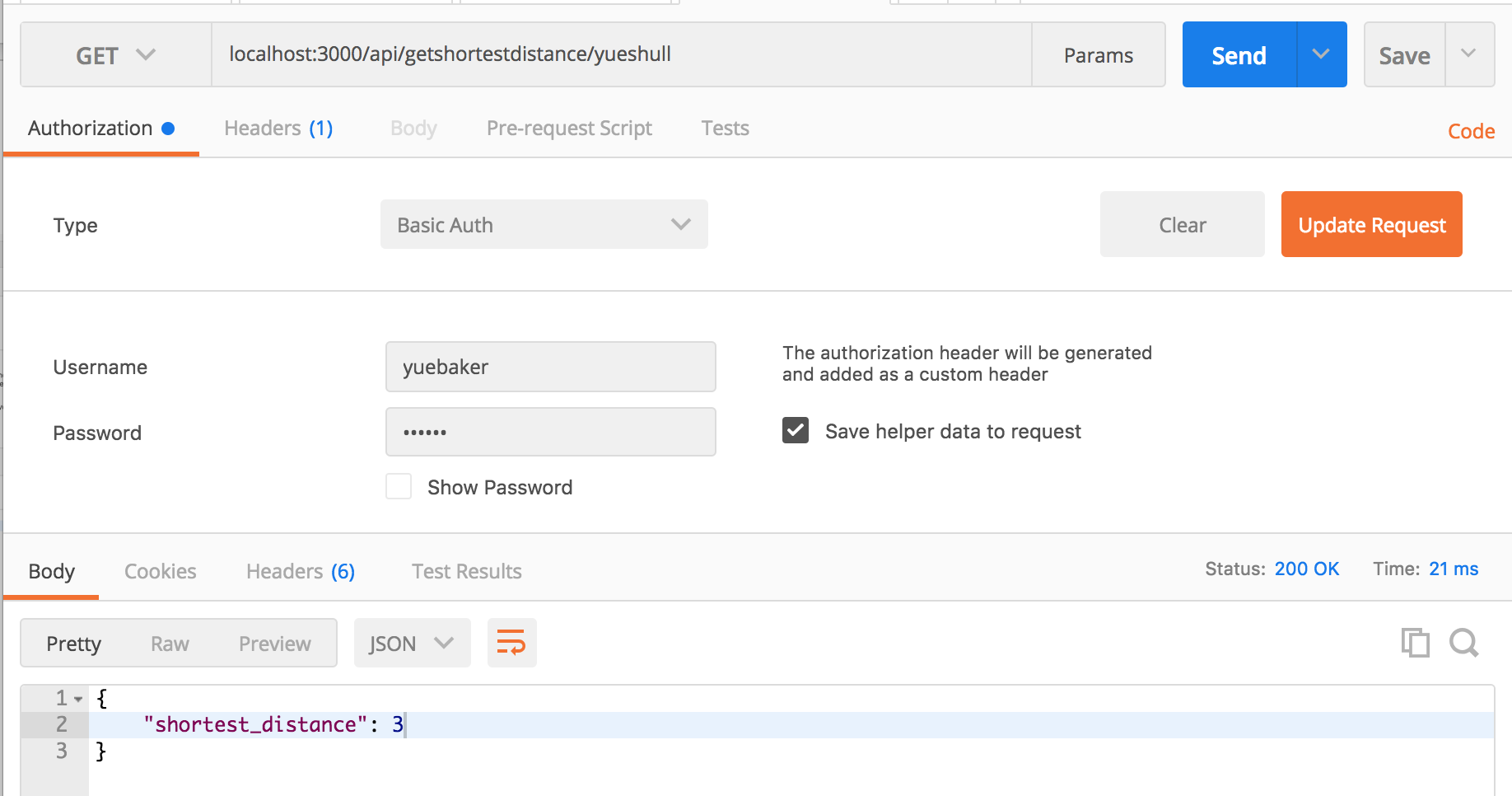


1. An endpoint that returns the current user's "shortest distance" to some other user. The shortest distance is defined as the number of hops needed to reach a user through the users you are following (not through your followers; direction matters). For example, if you follow user B, your shortest distance to B is 1. If you do not follow user B, but you do follow user C who follows user B, your shortest distance to B is 2.

This is the api call to get the shortest distance from the current user to the target user

'/getshortestdistance/:username'





Solution Algorithm: N-ary Tree structure BFS algorithm (getfollowings to get the children nodes to each of the root node)

yuebaker

(root)

fengy nikolab chirsh

yueshul

yueshull